SIP Message Overflow Routing Scheme (SMORS)

Alexander A. Kist and Richard J. Harris

RMIT University Melbourne
BOX 2476V, Victoria 3001, Australia
Telephone: (+) 61 (3) 9925-5218, Fax: (+) 61 (3) 9925-3748
Email: {kist,richard}@catt.rmit.edu.au

***Abstract*— The 3rd Generation Partnership Project (3GPP) will use the Session Initiation Protocol (SIP) as a session signalling protocol in the IP Multimedia Subsystem (IMS) of next generation UMTS systems. Certain nodes on the signalling layer have to process a large number of SIP messages. This paper introduces a concept for efficient SIP message routing and user assignment. The SIP Message Overflow Routing Scheme (SMORS) scheme is defined that has two major advantages over existing persistent hash routing: The content or user location in SMORS is arbitrary and it provides overflow and backup routing capabilities.**

I. Introduction

The Session Initiation Protocol (SIP) is defined in RFC 3261 [1] and performs user location, session setup and session management. The 3rd Generation Partnership Project (3GPP) is a global initiative to develop standards and specifications for next generation *Universal Mobile Telecommunications System* (UMTS) networks. 3GPP has decided to use SIP as the signalling protocol for the IP Multimedia Subsystem (IMS) and introduces a number of SIP proxy servers called *Call Session Control Function* (CSCF). These servers are used by Commercial service providers to control session signalling message flows and enable authentication, billing and service provisioning. 3GPP Technical Specification 23.228 [2] (R5) explains these functions in more detail.

Logically, SIP nodes are located on the application layer and can be connected by virtual SIP Links (VSLs) [3]. VSLs and SIP nodes form *Virtual SIP Overlay Networks* (VSONs). Messages traversing VSONs can take alternative routes. In 3GPP IMS most routing decisions are made during the registration of users since intermediate nodes record registration state and/or session state information.

3GPP uses several different SIP proxy server functions to perform specific tasks. *Proxy* CSCFs (P-CSCFs) are the network entry points for the *User Equipment* (UE). *Serving CSCFs* (S-CSCFs) hold a copy of the user profile, record session state information and provide higher level session handling functions. *Interrogating CSCFs* (I-CSCFs) are network entry points for terminating sessions and decide the message routing to S-CSCFs. I-CSCFs have to route SIP messages with minimal (state) information, and since they serve a large number of requests, messages have to be served as efficient as possible. Usually, the associations between users and S-CSCFs in the I-CSCF are set during registration of users and subsequent messages have to be routed through the same nodes.

SIP message routing in 3GPP can be divided into two areas: Routing decisions have to be made for routes from I-CSCFs to S-CSCFs and routing decisions are required for routes from UEs/P-CSCFs/S-CSCFs to I-CSCFs. Where the first requires methods that resemble load balancing schemes, the latter can be solved by shortest path routing methodologies [4].

The challenge of efficient message routing is similar to existing server/resource allocation problems which include server load balancing, hash routing and web caching schemes. In recent years, these areas have been major research targets. Barish and Obraczka discuss in [5] trends and techniques in web caching and Ross introduces in [6] classical hash based routing to allow load balancing in web server pools. Robust hashing or persistent hashing, such as *Highest Random Weight* (HRW) mapping [7], calculates hash values over the object addresses and servers where the object is located, so rearranging the hash space does not disrupt the assignments. The major focus of these efforts relates to issues concerning the location of distributed document sets, such as web pages that consist of several files. The article by Kencl and Le Boudec [8] uses robust hashing to implement an adaptive load sharing scheme for network processors.

The principal concept of overflow routing is not new and has been extensively studied in the context of *Public Switched Telephone Networks* (PSTNs). Gerald Ash's book [9] presents a comprehensive discussion of this topic. It includes examples such as *Dynamic Non-Hierarchical Routing* (DNHR) which uses different path sets for different times of the day for voice carriers and was initially developed by AT&T. The scheme that is proposed in this paper, is based on the *Scheme for Alternative Packet Overflow Routing* (SAPOR) [10] which uses similar methodologies for IP packet routing.

This paper introduces the *SIP Message Overflow Routing Scheme* (SMORS). It targets efficient message routing for high volume SIP servers, in the 3GPP IMSs context and it specifically addresses message routing between I-CSCFs and S-CSCFs. The contribution in this paper is twofold: Firstly a concept for efficient SIP message routing and user assignment is introduced, and secondly, the SMORS hash routing scheme is defined. SMORS has several benefits: It is fast and requires little processing for subsequent message routing, it has load balancing, backup provisioning and overflow routing capabilities. One major advantage of this scheme is that the user-to-S-

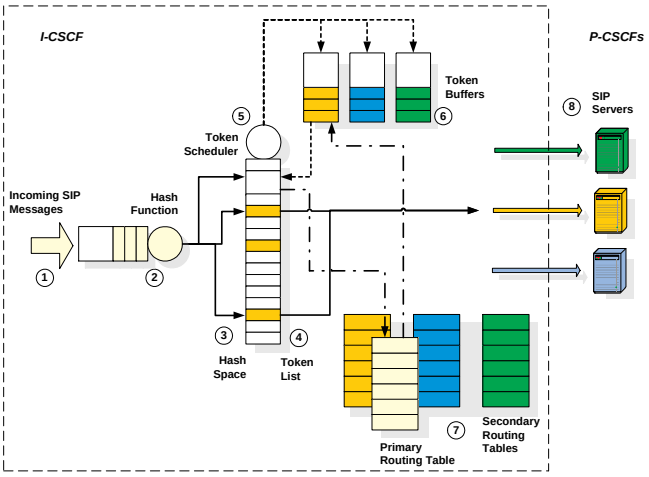

Fig. 1. SMORS Scheme

CSCF assignment is flexible. S-CSCFs can be assigned, by user priority, service subscription or alphabetically by user name. The assignments do not have to be calculated at runtime. These associations can be pre-calculated.

This paper is organised as follows: Section II defines SMORS, Section III discusses issues concerning the extension of SMORS to multiple servers, Section IV elaborates on the different functional components. The paper concludes with remarks on requirements, performance and operation of SMORS in Section V.

II. Concept

Certain SIP servers have to route large numbers of messages as quickly as possible. Discussions above outlined that I-CSCF's act as network entry points for SIP messages in the 3GPP IMS. A primary process is required that matches messages and their appropriate S-CSCFs. To use SMORS overflow routing, servers require a SMORS subfunction.

Figure 1 depicts the schema of the SMORS function. The main functional groups are the *Hash Function*, the *Token System* and the *Routing Tables*. The hash function consists of two parts: the actual hash function (2) and the hash space (3). The token system consists of the token buffers (6), the token scheduler (5), and the token list (4), which is equivalent to the hash space (3). The routing tables (7) are a selection of different tables. Incoming messages are accumulated (1), and routed to the servers (8). The dotted lines indicate the token flows, the dash-dotted lines indicate requests to the routing tables, and the solid lines show the virtual message flows in this scheme. Every server has an associated token buffer. To distinguish the different servers and buffers, colours (shading) are used to indicate their affiliation.

Incoming messages (1) are processed by the hash function (2) and are assigned to a (unique) hash space (3). If the corresponding token space in the token list (4) holds a token, the message is directly routed to the appropriate server (8). If the token space is empty, the primary routing table (7) is consulted. The tables have entries or rules that allow the selection of servers. Once the server is known, a token is taken from the appropriate token buffer (6) and assigned to the token list. This reduces the number of tokens in the token buffer by one. If no more tokens are available in the token buffer, the secondary routing table is consulted and the same process is repeated. The actual SIP server message processing is located before and/or after the routing tables are consulted. Message overflow occurs when the token buffer of one server is empty. Existing assignments remain, however additional users are routed to alternative servers, specified in the secondary routing tables. Tokens in SMORS represent available users-spaces in servers. If new messages arrive that map onto the same hash space which already holds a token, the messages are sent to the appropriate server. Tokens have a finite time to live. The lifetime is longer than the registration period. If a token space is not visited by a register message during this period, it is cleared and returned to the appropriate token buffer by the token scheduler (5). If a register-message arrives before the timer has expired, it is reset.

Usually the process of user-to-server assignment is executed during registration, but SMORS triggers routing decisions when the first message of a particular user is received. In this case, the token list is empty. Under normal operating conditions, this occurs at the time of the first registration. This concept is basic and practical implementations will consist of a large number of processing functions (servers). Therefore, the next section explains a scheme that takes several server nodes into account and has no single point of failure.

III. SMORS Farm

In the case that I-CSCFs have to serve a large number of users, several servers that belong to the same I-CSCF (server farm), serve multiple requests simultaneously. Here, a more elaborate configuration with several SMORS functions is required. Figure 2 depicts a scheme that addresses the problem. It consists of a number of possible gateway nodes (1) that perform persistent hashing (2) to assign the messages to a SMORS function (3). To divide the users between n SMORS functions, the hash space is divided into n parts and every part is mapped to a SMORS function. The SMORS functions operate as described in Section II and route the messages to the appropriate S-CSCFs (5).

Since robust hashing is used, the SMORS-to-user assignment remains the same, even if messages use different entry points. They are routed to the same SMORS subfunction. New server subfunctions can be added to increase the capacity and response time of the I-CSCF. The only information that has to be shared between the different SMORS functions is the number of available S-CSCF users spaces, i.e. the number of tokens in buffers. The *Update Buffer Function* (4) equalises the number of tokens in the different SMORS components after an update interval. For practical implementations, this function requires a backup function. It can also be used by a feedback system where the S-CSCFs inform the *Update Buffer Function* about their capabilities. The next section discusses the SMORS components in more detail.

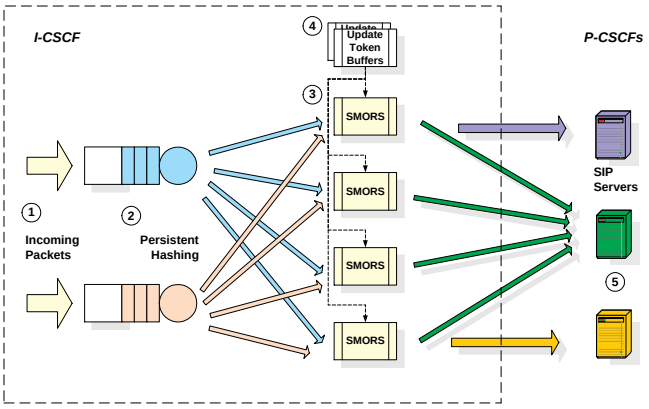

Fig. 2. SMORS Farm

IV. Building Blocks

SMORS uses three major building blocks: The hash function, the token system and the routing tables. This section discusses the functional components in more detail.

A. Hash Function

The purpose of the hash function is to separate single users on the basis of message specific parameters such as the source IP address, User ID and other parameters. A wide range of possible hash functions exist. The work by Cao et al. [11] discusses hashing based schemes for Internet load balancing. The important attributes of a hashing scheme for SMORS can be summarised as follows: The results have to be evenly distributed over the hash space and, if possible, the hash space should be large enough to ensure that no frequent overlapping occurs. A 16 bit hash function, for example, yields $65,536$ hash spaces. For the purpose of SMORS, the XOR folding of the source (user ID) address appears to be sufficient. Equation (1) shows this calculation.

$$H(.) \quad = \quad \Big(S_1 \oplus S_2 \oplus S_3 \oplus S_4\Big) \bmod N \qquad (1)$$

The ith byte of the source IP address is denoted by S_i. N limits the size of the hash space. [11] indicates a sufficient spreading and this scheme is simple to implement. Collisions have to be accounted for by a secondary data structure such as linked lists etc. Alternative hash functions can also be used.

If the mapping of the hash space changes, disruptions can occur since the mapping has to be reorganised. The concept of robust hashing [7] combats this setback. However, this problem does not occur for SMORS, since the token mapping is persistent, even when new servers are added or they disappear. However SMORS farms use robust hashing at the ingress to split users between several SMORS functions. In this case it is required that existing associations between parts of the hash space and single SMORS functions encounter a minimal number of disruptions if the topology changes. The next section discusses the token system.

B. Token System

The token system is a major part of the SMORS scheme. The number of available tokens defines the number of possible users in destination servers. Tokens are used in a similar way to the token buffers, used for the leaky bucket scheme (e.g. [12]). A token buffer is associated with a server and has a limited number of tokens. Every time a user is registered at a server, a token is assigned to the destination. If no tokens are available, no new users can be routed to this server. Tokens have a (token) time to live (TTTL). If a server receives no more register messages and the TTTL has expired, the tokens are returned to the appropriate buffer. In this case, I-CSCFs record a minimal amount of registration state information: Tokens identify S-CSCF; they are quasi-state information. The token system components are:

1) Token Buffers: Token buffers can be implemented by a counter that denotes the number of tokens in a buffer. If a token is removed, the number decreases. If a token is returned, the number increases. Under normal operating conditions, this number is positive, but if the bucket size changes during operation, the bucket count can be negative. No tokens are available if the token count is less than one.

2) Number of Tokens: The major parameter that specifies a token buffer is the number of available tokens ν. It defines the maximum number of users in the associated server. The calculation, shown in this section, assumes that the average user number is a valid approximation. The registration period is denoted by $\overline{t}$ and is measured in seconds. The user arrival rate is denoted by ϑ and measured per second. The number of active users A can be calculated by Equation (2).

$$A = \vartheta \cdot \overline{t} \qquad (2)$$

Equation (3) shows the calculation of the number of required tokens ν_s for server s. A_s is the capacity of server s in users. The second sum calculates the number of tokens that are required due to the TTTL of the tokens and the delayed reset of τ.

$$\nu_s = A_s + \vartheta \cdot \tau \cdot \frac{A_s}{\sum_{i=1}^{s_{max}} A_i} \qquad (3)$$

where s_{max} is the number of servers. The second product is necessary to scale the rate.

If SMORS farms are used, the number has to be divided by the number of possible alternative SMORS functions, and security margins have to be added to account for statistical effects.

3) Token Scheduler: The token scheduler is the function that determines "expired tokens", clears the token list and returns them to the token buffers. The token scheduler is the most expensive function of the SMORS scheme, but it is not frequently used. It requires two major sub functions: The traversing of the token list and the implementation of the TTTL.

The token list can be implemented by using a combined array-linked-list data structure. This avoids the requirement that all spaces have to be traversed for the price of

additional memory. The TTTL can be implemented with a simple counter variable. Every token list item has one such variable associated with it. Zero indicates that the space is empty, i.e. no messages are using this space. Empty spaces are skipped by the iteration operation. If the number is positive it is increased every time it is traversed by the iteration function. If a register message arrives during the update interval, the number is reset by this event. The iteration function is executed every T_c seconds. If n reaches the maximum count n_{max}, the counter is reset to zero and the token is returned to the appropriate token buffer. n_{max} is a positive integer which is larger than one.

4) Token Update Interval: Since the token scheduler is the most expensive function, the token update interval T_c should be as long as possible. On the other hand, long intervals require more tokens, a larger token list and the traversing of more active tokens in the token list. The number should be chosen on the basis of the evaluation of the given constraints.

C. Routing Tables

The function that routes the different users utilises a number of alternative routing tables. They are denoted by primary, secondary, tertiary, ... n-ary table etc. These tables specify associations between servers and users and they can be generated by any means that are appropriate and useful for the network configuration. The process where these associations are formed is not in the scope of this work.

In practice, the routing tables will use similar criteria to the ones that are discussed in [4] and consider operator policies, delay, reliability, etc. Other criteria can include user (profile) specific information such as user class, subscription type, frequency of use etc. SMORS uses one primary routing table per node and subsequent routing tables can be S-CSCF specific. The routing decisions for the n-ary routing tables can be based on any factors, e.g. administrative decisions by human network operators.

Secondary routing tables could also define a complete redundant server system, only used if the primary system is overloaded or fails. Existing teletraffic engineering methodologies can be applied to optimise the performance and the reliability of the server system.

V. Remarks

A. Simple Admission Control

SMORS can be used to implement a simple first barrier security check: If the token list has an additional mark that indicates valid users, requests are only served if their addresses map to a valid token space otherwise they are ignored. This does not provide a high level of security, but reduces the processing which is required to identify invalid requests.

B. Overflow Users

In practical applications users will be registered for periods that might extend periods with high user volumes and overflow situations. If no additional conditions are implemented, these users will remain in the overflow nodes, even when capacities in the primary servers become available. This can be resolved if, in the case of empty spaces in the primary server, a reroute is requested during re-registration. In this way the users will be moved to the primary server and the overflow servers are vacated. Users with long registration periods will be placed in the primary servers.

C. Topology Changes

1) SMORS: In the case of server topology changes, the scheme has to adapt. When new servers come online, SMORS will simply shift users onto new server capacities as they become available. If a server is failing, the corresponding token buffer is emptied and all tokens in the token list that belong to this server are flushed. In this way, all existing users will be assigned to new servers.

2) SMORS Farm: The SMORS farm is designed in a way that it has no single point of failure. The configuration minimises the disruption if one of the functions fails. The first nodes that can fail are the persistent hashing functions. Since these functions perform exactly the same tasks and record no state information, they can simply be substituted for each other. The next function that can fail is one of the SMORS functions. In this case, all messages that were routed using this SMORS function have to be reassigned. This is done by the persistent hashing function block. Under normal load conditions without any overflow users, the users will be assigned to the same server as before and no disruption occurs.

If overflow users were routed to other servers this might cause a marginal disruption. All users that were served by the failed SMORS function are reassigned, in the order which new messages arrive to these nodes. The first arrivals are routed to the primary S-CSCF, later arrivals might be routed to the overflow S-CSCF. Since this order may be different from the original order, users that were located in the primary S-CSCF before, are now located in the overflow S-CSCF and vice versa. In these cases, the S-CSCFs have to reload the user specific data, i.e. user profiles etc. Since this process concerns only a small fraction of users, it is very limited and only causes additional delays in the S-CSCF message processing. The details of this process depend on the exact implementation of the registration function. If S-CSCF servers fail, the overflow capability of SMORS functions is used: The token buffer which corresponds to the failing server, is flushed and messages are rerouted to new S-CSCFs.

D. Requirements and Performance

1) Complexity: The complexity of SMORS is twofold. All operations that are required during message routing are of the order of one: $O(1)$ i.e. the hash function, the check if a token is assigned, the lookup of the routing table, the assignment of a new token and the changes in the token buffer, therefore, it is possible that all messages are routed in $O(1)$ time. If a token is already assigned to the buffer, fewer $O(1)$ steps are required. The second operation is more complex but it is less frequently required. To clear and update the token list, all spaces have to be traversed.

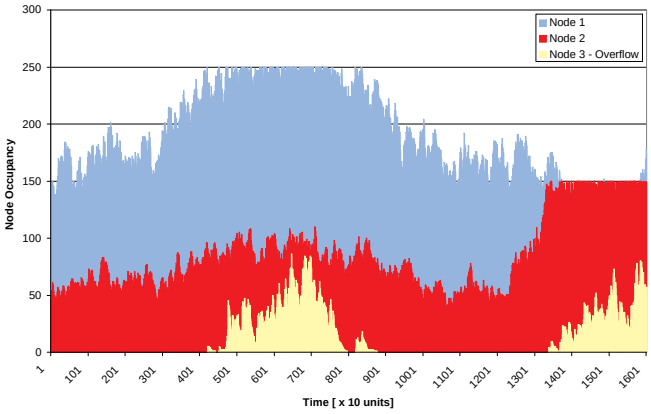

Fig. 3. Node Occupancy

The complexity of this process is therefore $O(l)$, where l indicates the size of the token list and the hash space. Using more memory and intelligent data structures can reduce the number of spaces that have to be traversed.

2) Memory: The memory requirements for the routing tables are equivalent to the requirements for tables of legacy systems. The token buffers are simple counters and require only several bytes per buffer. The token list is of size l and requires l bytes. However, a more advanced token list requires $k \cdot l$ bytes.

VI. Simulation Results

This section introduces simulation results that show the operation of SMORS. A discrete event simulator was used which utilises the Mersenne Twister [13] with a period of $2^{19937-1}$ as a random number generator. A topology with three SIP servers was simulated. The users were split between two nodes (Node 1 and Node 2) on the basis of arbitrary address values. Overflow users were routed to a third node (Node 3). The registration requests in this simulation followed a Poisson arrival process, with an initial mean arrival rate of 1.5 users per time unit for users destined for Node 1 and 0.5 users per time unit for users destined for Node 2. The registration period was exponentially distributed with a mean of 100 time units. The capacity of the servers was set to be 250 users for Node 1, 150 for Node 2 and 200 for Node 3. During the simulation, the arrival rate was doubled for sessions destined for Node 1 and towards the end of the simulation the session arrivals of register requests destined for Node 2 were strongly increased.

Figure 3 depicts the simulation results. The occupancy of Node 1, Node 2 and Node 3 were measured every 10 time units. The graph depicts the node occupancy versus time. At time 300 it shows a increase of users in Node 1. When the server limit of 250 users is reached, the overflow users were routed to Node 3. This is reflected in the increase in occupancy in Node 3 from zero to about 50 users. As the users, destined for Node 1 decrease, Node 3 is eventually vacated. Towards the end of the simulation period, at time 1300, the users destined for Node 2 are increased. If the threshold of 150 users is reached the overflow users are again routed to Node 3. The simulation showed the overflow capability of SMORS with two primary and one overflow server.

VII. Conclusion

This paper defined the SMORS scheme allowing efficient SIP message routing in high volume 3GPP SIP I-CSCF servers. Routing decisions are persistent for all requests that have the same local destination. Further work has to consider the adaptation of the scheme to practical implementations and configurations, so further performance evaluations become possible. Exact routing decisions have to be further investigated once more specific implementation details become available.

SMORS can use redundant servers to reduce the disruption in cases of node failures and additional resources can be added without disruption of the existing users-assignment. SMORS is a highly flexible framework that allows for extensions to address specific emerging challenges. The concept that was introduced in this paper can also be adapted to other scenarios. Examples include distributed caching systems and load sharing in between different processors.

Acknowledgements

The authors would like to thank the Australian Telecommunications Cooperative Research Centre (ATcrc) for their financial assistance for this work.

References

[1] J.Rosenberg, H.Schulzrinne, G.Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, IETF, June 2002, RFC 3261 (Obsoletes: RFC 2543).

[2] 3rd Generation Partnership Project, *IP Multimedia (IM) Subsystem - Stage 2 (Release 5)*, July 2001, 3GPP TS 23.228 V5.1.0.

[3] A.A. Kist and R.J. Harris, "Using virtual SIP links to enable QoS for signalling," *In ICON 2003, Sydney, Australia*, September 2003, To appear.

[4] A.A. Kist and R.J. Harris, "SIP routing methodologies in 3GPP," *In First International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs '03), Ilkley, West Yorkshire, U.K*, July 2003.

[5] G. Barish and K. Obraczka, "World wide web caching: Trends and techniques," *IEEE Commun. Mag.*, pp. 178–184, May 2000.

[6] K.W. Ross, "Hash-routing for collections of shared web caches," *IEEE Network Magazine*, pp. 37 – 44, Nov-Dec 1997.

[7] David G. Thaler and Chinya V. Ravishankar, "Using name-based mappings to increase hit rates," *IEEE/ACM Trans. Networking*, pp. 1 – 14, February 1998.

[8] Lukas Kencl and J. Y. Le Boudec, "Adaptive load sharing for network processors," *In Infocom 2002*, June 2002.

[9] G. R. Ash, *Dynamic Routing in Telecommunication Networks*, McGraw-Hill, 1997.

[10] A.A. Kist and R.J. Harris, *Scheme for Alternative Packet Overflow Routing (SAPOR)*, In Workshop on High Performance Switching and Routing (HPSR 2003), Torino, Italy, June 2003.

[11] Z. Cao, Z. Wang, and E. Zegura, "Performance of hashing-based schemes for internet load balancing," *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2000*, vol. 1, pp. 332 –341, 2000.

[12] E. P. Rathgeb, "Modeling and performance comparison of policing mechanisms for ATM network," *IEEE J. Select. Areas Commun.*, vol. 9, no. 3, pp. 325–334, April 1991.

[13] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM Trans. on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, January 1998.